Burak Tekdamar

161044115

CSE 344 HW4 REPORT

1 Overview

First of all, to solve the problem, I read the parameters given by the user one by one with the getopt() function and assigned them to the variables. I checked whether the parameters given by the user are valid. I returned an error message if there is an invalid parameter or a missing parameter. I created functions named consumer and supplier that threads will use. The supplier loop works to read one character at a time from the input file. The consumer loop runs as much as the N value entered by the user. consumers will wait for the semaphores and the supplier will post to these semaphores as they read characters from the file. If the '1' character is read, the first semaphore will be posted, if the 2 character is read, it will be posted to the second semaphore. Thread of supplier is set to be detachable. At the end of the program, it will wait for all threads to terminate and will terminate.

2 How did I solve this

First of all, I kept a pointer to hold consumer threads. I converted this pointer to a dynamic array with the C value from the user. Since the supplier thread will be detachable, I gave an attr parameter with the PTHREAD_CREATE_DETACHED parameter while creating the thread. Then I created consumer threads in a loop. I have passed the id of each thread as a parameter to the function. I joined consumer threads to wait for all consumer threads to terminate. I have created 2 separate functions called waitSem and postSem to control semaphores. waitSem takes semaphore and two separate parameters. one for the '1' character and the other for the '2' character. If neither of these two characters is posted, the consumer thread cannot exit wait. The postSem function takes these characters separately.

```c
void waitSem(int sem, unsigned short a, unsigned short b) {
    struct sembuf sop[2];
    sop[0].sem_num = a;
    sop[0].sem_op = -1;
    sop[0].sem_flg = 0;
    sop[1].sem_num = b;
    sop[1].sem_op = -1;
    sop[1].sem_flg = 0;

    if(semop(sem, sop, 2) == -1) {
        fprintf(stderr, "waitSem semop error");
        exit(EXIT_FAILURE);
    }
}
```

waitSem function

```c
void postSem(int sem, unsigned short n) {
    struct sembuf sop;
    sop.sem_num = n;
    sop.sem_op = 1;
    sop.sem_flg = 0;
    if(semop(sem, &sop, 1) == -1) {
        fprintf(stderr, "postSem semop error");
        exit(EXIT_FAILURE);
    }
}
```

postSem function

Supplier thread reads only 1 character at a time from the file. If this character is '1' it posts semaphore number 0, if this character is '2' it posts semaphore number 1. When the end of the input file is reached, the supplier thread terminates. All of the consumer threads are running as much as the N value entered by the user. These consumer threads are putting semaphores 0 and 1 on hold at the same time. they continue to work when posted in two semaphores. These threads terminate when the number of iterations reaches N. When the threads are terminated, the program closes the semaphores, frees the consumer pointer and closes the input file.

Valgrind Result:

```
==390282== HEAP SUMMARY:
==390282==     in use at exit: 0 bytes in 0 blocks
==390282==   total heap usage: 18 allocs, 18 frees, 5,806 bytes allocated
==390282==
==390282== All heap blocks were freed -- no leaks are possible
==390282==
```

Running hw4 program:

./hw4 -C consumerNum -N iterationNum -F inputfilePath

I wrote a handler to catch the SIGINT signal. When the user presses the CTRL+C, I set the atomic control variable to FALSE to terminate the threads. Then it closes the semaphores, frees the consumer pointer, closes the input file and ensures that the program terminates properly. When CTRL C is pressed, a possible leak appears in valgrind as much as 272 byte * threads due to the pthread_create function.

```
==390029== LEAK SUMMARY:
==390029==    definitely lost: 0 bytes in 0 blocks
==390029==    indirectly lost: 0 bytes in 0 blocks
==390029==      possibly lost: 2,992 bytes in 11 blocks
==390029==    still reachable: 0 bytes in 0 blocks
==390029==         suppressed: 0 bytes in 0 blocks
==390029==
```

OUTPUTS

```
1652481090 Consumer-3 at iteration 9015 (consumed). Post-consumtion amounts: 1 x '1', 0 x '2'
1652481090 Consumer-3 at iteration 9016 (waiting). Current amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
1652481090 Consumer-5 at iteration 9057 (waiting). Current amounts: 0 x '1', 0 x '2'
1652481090 Consumer-7 at iteration 9048 (consumed). Post-consumtion amounts: 1 x '1', 0 x '2'
1652481090 Consumer-7 at iteration 9049 (waiting). Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
1652481090 Consumer-4 at iteration 9059 (consumed). Post-consumtion amounts: 0 x '1', 0 x '2'
1652481090 Consumer-8 at iteration 9035 (consumed). Post-consumtion amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
1652481090 Consumer-4 at iteration 9060 (waiting). Current amounts: 1 x '1', 0 x '2'
1652481090 Consumer-8 at iteration 9036 (waiting). Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
1652481090 Consumer-9 at iteration 9058 (consumed). Post-consumtion amounts: 0 x '1', 0 x '2'
1652481090 Consumer-9 at iteration 9059 (waiting). Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
1652481090 Consumer-1 at iteration 9055 (consumed). Post-consumtion amounts: 0 x '1', 0 x '2'
1652481090 Consumer-1 at iteration 9056 (waiting). Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Consumer-2 at iteration 9060 (consumed). Post-consumtion amounts: 0 x '1', 0 x '2'
1652481090 Consumer-2 at iteration 9061 (waiting). Current amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 1 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 1 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 0 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 0 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 1 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 1 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 2 x '2'
1652481090 Consumer-6 at iteration 9058 (consumed). Post-consumtion amounts: 0 x '1', 2 x '2'
1652481090 Consumer-6 at iteration 9059 (waiting). Current amounts: 0 x '1', 2 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 2 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 3 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 3 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 4 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 4 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 5 x '2'
```

```
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 2 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 2 x '2'
1652481090 Consumer-3 at iteration 9206 (consumed). Post-consumtion amounts: 0 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9207 (waiting). Current amounts: 0 x '1', 2 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 2 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 3 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 3 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 4 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 4 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 5 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 0 x '1', 5 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 6 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 6 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 5 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 5 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 5 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 1 x '1', 5 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 2 x '1', 5 x '2'
1652481090 Consumer-3 at iteration 9207 (consumed). Post-consumtion amounts: 0 x '1', 5 x '2'
1652481090 Consumer-3 at iteration 9208 (waiting). Current amounts: 2 x '1', 5 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 2 x '1', 5 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 2 x '1', 4 x '2'
1652481090 Consumer-3 at iteration 9208 (consumed). Post-consumtion amounts: 1 x '1', 4 x '2'
1652481090 Consumer-3 at iteration 9209 (waiting). Current amounts: 2 x '1', 4 x '2'
1652481090 Consumer-3 at iteration 9209 (consumed). Post-consumtion amounts: 1 x '1', 3 x '2'
1652481090 Consumer-3 at iteration 9210 (waiting). Current amounts: 1 x '1', 3 x '2'
1652481090 Consumer-3 at iteration 9210 (consumed). Post-consumtion amounts: 0 x '1', 2 x '2'
1652481090 Consumer-3 at iteration 9211 (waiting). Current amounts: 0 x '1', 2 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 2 x '1', 4 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 1 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 0 x '1', 1 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 1 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 1 x '1', 1 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 1 x '1', 2 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 1 x '1', 2 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 2 x '1', 2 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 2 x '1', 2 x '2'
1652481090 Consumer-3 at iteration 9211 (consumed). Post-consumtion amounts: 0 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9212 (waiting). Current amounts: 3 x '1', 2 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 3 x '1', 2 x '2'
1652481090 Supplier: read from input a '1'. Current amounts: 2 x '1', 1 x '2'
1652481090 Supplier: delivered a '1'. Post-delivery amounts: 3 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9212 (consumed). Post-consumtion amounts: 2 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9213 (waiting). Current amounts: 3 x '1', 1 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 3 x '1', 1 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 2 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9213 (consumed). Post-consumtion amounts: 2 x '1', 0 x '2'
1652481090 Consumer-3 at iteration 9214 (waiting). Current amounts: 2 x '1', 1 x '2'
1652481090 Supplier: read from input a '2'. Current amounts: 2 x '1', 1 x '2'
1652481090 Supplier: delivered a '2'. Post-delivery amounts: 1 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9214 (consumed). Post-consumtion amounts: 1 x '1', 0 x '2'
1652481090 Consumer-3 at iteration 9215 (waiting). Current amounts: 1 x '1', 1 x '2'
1652481090 Consumer-3 at iteration 9215 (consumed). Post-consumtion amounts: 0 x '1', 0 x '2'
1652481090 Consumer-3 has left
```


1212
1212
1212
1212
1212
1212
1212
1212
1212
1212
2122
2222
1111
1121
1122
1212
1212
1212
1212
1212
1212
1212
1212
1212
2122
2222
1111
1121
1122
1212
1212
1212
1212
1212
1212
1212
1212
1212
1212
2122
2222
1111
1121
1122
1212
1212
1212
1212
1212
1212
1212

Input file